Model−Based Battle Command: A Paradigm Whose Time Has Come

Samuel C. Chamberlain

Army Research Laboratory
Aberdeen Proving Ground, MD 21005−5067

ABSTRACT

This paper presents a regenerated perspective of the way battle command is automated that has been enabled by the amazing increase in computing performance now realized in small packages. The goal is to fully exploit this power while simultaneously addressing other persistent problems, such as interoperability, integration, tenuous communications, and cost−effective implementation. A paradigm coined *model−based battle command* is introduced that represents a culmination of related, general concepts addressed at this and other symposiums over the past decade.

1. Introduction

In spite of the incredible advances in computing, the automation of battle command remains relatively antiquated because of the common pitfall of simply automating manual techniques. Computer processing power is advancing at a rapid pace with an equally impressive decrease in cost. For the first time, the reduced size, weight, and power requirements of hardware allow sophisticated computing capabilities to be available at any military echelon and environment, to include the individual warrior. Yet, in spite of this power, battle command automation remains focused on *communications−intensive, message−based* paradigms.

This paper describes a different perspective for automating battle command systems that is centered on truly exploiting the multiplying, available computational power through *model−based* (rather than message−based), *computationally intensive* paradigms of battle command. This approach will be intuitive and obvious to most that attend this 1995 Symposium on C2 Research and Technology and has indeed been inferred, or implied, in numerous papers on related topics. However, it has never been directly and explicitly stated in spite of several supporting research and standardization programs. Therefore, this paper will address model−based battle command as both a trend and a basic philosophy for building battle command systems.

Battle command system developers have traditionally included modeling and simulation (M&S) as a major element of their acquisition strategy (in fact, it is now mandated). But it is interesting and bothersome that battle command is viewed so differently between the system development and M&S communities. A major disparity exists between the way we *do* battle command and the way we *model* battle command. Although numerous research programs have produced a diverse set of battle command models and simulations, they are still relegated to a role of support and study rather than being considered as part of the actual system. Under a model−based paradigm of battle command, the system development and M&S communities unite so that using automated tools to *do* battle command is merely *simulating* battle command with real−time, real−world data. Simply stated, from an automation perspective: modeling battle command should differ little from actually ‘‘doing’’ battle command.

2. Message−Based Battle Command

2.1 Procedural Messages

To better understand model−based battle command we must first discuss the traditional, message−based scheme. Before digital computers and communications were readily available, voice−based *procedures* were defined to facilitate information exchange and assimilation between human operators. With the advent of computers, voice moved to text. Thus, the voice procedures moved to digital text procedures that were implemented via sets of messages (hence the term: *procedural messages*).

2.2 Three Stages of Information Exchange

Figure 1 illustrates three stages of information exchange in an automated environment. First, the human enters information into the computer; second, the information is exchanged between the computers; and third, the information is retrieved (or presented) to the human. With the digital Teletype, all three stages used the same form: text messages. As expected, message comprehension quickly became a problem, and policy was developed to standardize message sets to follow

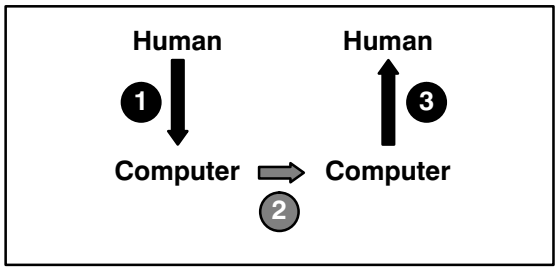

Figure 1: Three Stages of Information Exchange

the procedures carried over from the voice domain. As automation increased, machines began to extract data from the standard message formats. This process has continued to evolve to a sophisticated and expensive level.

Overlooked is the fact that the exchange of information between computers (i.e., stage 2) need not mimic the exchange of information between humans and computers (i.e., stages 1 and 3, resp.). As a result, standardized messages intended for the exchange of information between computers actually contain numerous characteristics that reflect human−computer interface details rather than computer−computer details.

Simple examples are time and position. There are numerous variations of these entities because of differences in human cognition of resolution and units. For time, there are standard entities based on whether second, minute, hour, or day (i.e., date) resolution is required. To humans, date and time are considered different entities. But computers do not make this distinction internally anymore. These are human−to−computer (and vice versa) issues that can be handled with a single, general data structure between computers, i.e., a value and a resolution indicator. For example, a ''date'' can be stored as the number of seconds since 0000 GMT, 1 January 1980 until noon of the desired date plus or minus 12 hours (i.e., a time with a resolution). The same is true for position data.

The point of this oversimplified example is that battle command data definitions are still viewed from a human−oriented, procedural perspective rather than from a computational, computer−oriented perspective. The right perspective must be used for the right purpose. This practice continues because (1) of a failure to recognize the fantastic processing power and ability of computers to convert between data formats and (2) it is a difficult labor and intellectually taxing process to define data abstractions of complex human concepts. Consequently, new message sets, based on traditional procedures, continue to be defined.[1]

1 A recent example is the Variable Message Format, or VMF, message set being designed by the Army for the Force XXI exercises. This includes messages for procedures such as an artillery Call for Fire and operational Spot, Situation, Contact, and Engagement Reports.

3. Model−Based Battle Command

In a model−based battle command scheme, the *model*, and its container, the database, become the center (rather than just a supporting resource) of the design and implementation of the automated portion of the battle command system.[2] Every unit in the force maintains its own copy of the battlefield model in its local computing environment.[3] This is now possible because of the amazing computational power and storage capacity available in small packages.

The database serves as the conduit, or hub, by which information is transferred between different units (i.e., nodes) and often, between the applications within the same unit. Currently, abstractions are manifested by ''information models'' (also called data models) that are often defined using tools that support a relational database perspective. For an excellent discussion of this approach see Bruce [1991].

The information in the model is stored in an abstract form conducive to machine manipulation rather than in a form suitable for direct human assimilation. However, human−oriented information, such as text, images, digitized audio, or graphic icons can be easily attached to the abstractions. But it is the abstractions that form the *backbone* of the battle command information infrastructure. Users rarely (if ever) directly access the data; instead, they rely on sophisticated, specifically tailored application programs, that in turn, access the database (or model). Thus, the user is completely unaware that an abstract model exists or that transactions have occurred.

3.1 Expanded Meaning of Communications

A model−based paradigm requires an expanded view of the task of communication. Abstract information is now exchanged *directly* between databases. To date, the tasks of databasing and communications have been independent, but now they must begin to merge. Communications, which was previously defined as the task of ''I send you a message and you send me a message,'' must become ''my database populates your database (data model) and your database populates mine.'' This merging of communications and databasing is a major change to the current way of doing business where the database is viewed an ancillary storage resource for messages or data elements. Now the database and its encapsulated model become the center of all automation activities. This is the essence of model−based battle command.

As a result, communication between models no longer need be accomplished via procedural messages that have traditionally been defined to ease creation and manipulation by human operators (e.g., an artillery Call For Fire or a maneuver Situation Report message). Now, models can exchange abstract information directly, even without the direct intervention

2 The term ''database'' is used as a general concept and does not imply a single, monolithic entity.

3 a ''unit'' being *any* element, i.e., down to a single warrior.

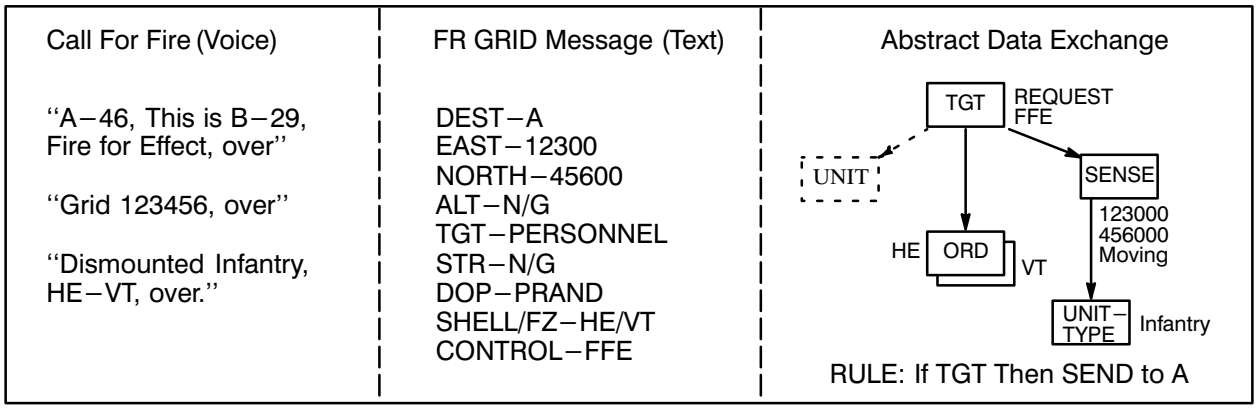

Figure 2: Voice, Message, and Model–Based Information Schemes

(or knowledge) of the end user. This requires breaking away from a traditional message–based approach of communications (i.e., an "e–mail" mentality) to one that is *transaction–based*. Figure 2 illustrates three different forms of an artillery Call for Fire. On the left is the sequence of voice messages that defines a Call for Fire procedure. In the middle is a digital text equivalent, a TACFIRE Fire Request Grid message. On the right is a pictorial view of a simplified data model of the meaning of a Call for Fire. In this case, new sensing and target entities are created that refer to each other and to existing reference information about ordnance and types of units. What is exchanged between nodes is a set of database commands similar to those used to update the local database. In other words, the data dictionary provides the structure for the messages.

3.2 Redirection of Standardization

A major side effect of a model–based battle command scheme is that the focus of standardization moves from that of standardizing messages (or data elements) to standardizing models. This is a significant redirection for a battle command system development community that has historically focused on standardizing the content of interchange rather than on the internal data storage schema. A relatively recent initiative now exists to develop a common "core" data model for command and control; see DISA [1994]. This project succeeds several data modeling efforts (see ATCCIS [1993]; Mayk [1993]; NRaD [1994]; and Walker [1993]), to include the JDL C2 Reference Model [Mayk, 1994]. However, this effort continues to exist in parallel with several major message and simulation standardization programs.

An emerging thesis is to focus on the similarities between models rather than their differences. An underlying belief of this author is that when viewed from an appropriate abstract level, all military forces, regardless of branch, service, function, or country are composed of the same basic entities and perform the same basic tasks — they just use different names. Consequently, the toughest challenge is the unification of these apparently disparate, but actually equivalent military entities and concepts. The traditional practice of automating manual techniques, which has been reinforced by the rush to standardize, continues to hinder progress in this key task. Caution must be exercised to avoid "vacuum–ware," which is software that is selected to fill a void simply because it is the first product to appear. Superficial data abstractions and standards only proliferate the problems of interoperability. A critical, comprehensive analysis by teams of military scientists and information systems designers is required to provide truly general data abstractions of military concepts.

3.3 Advantages of the Model–Based Paradigm

There are several major advantages to a model–based, computationally intensive approach to building battle command systems. First, the "backbone" information (i.e., the primary source of queries) is based on a formal model and is maintained in a form conducive to machine manipulation. Consequently, there is an innate expressiveness available to facilitate the automation of tasks and exploit the true computational power available.

For example, imagine that the data model includes a concept of a "sensing" — that is, something that is realized via the electromagnetic or acoustic spectrum. This could be a series of radar returns, a photograph, or a transcript of a report from a scout. Suppose that a human–oriented form of data is originally produced, such as an image. The image may be stored within a special "image server" tailored for such data. But the image would be tracked (or indexed) via a database that contains information in a machine–oriented form; in this case, via a "sensing" data abstraction; see Figure 3. The two data forms would be linked, and the image would be viewed as the "raw data" portioned of the "sensing" abstraction.

When first created, the sensing might include an "image title" and "date–time" information. Later, an analyst may study the image and populate the sensing abstraction with detailed, symbolic information derived from the image. It is this derived information that can be used by the computer to assist the commander in the decision process. Eventually, the analyst may decide that all necessary information has been derived from

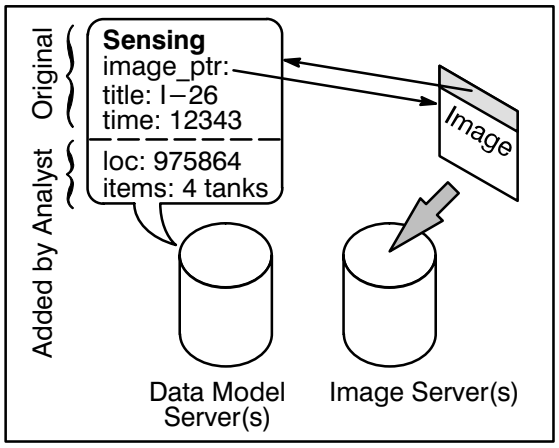

Figure 3: Concurrent Maintenance of Both Computer and Human−Oriented Information

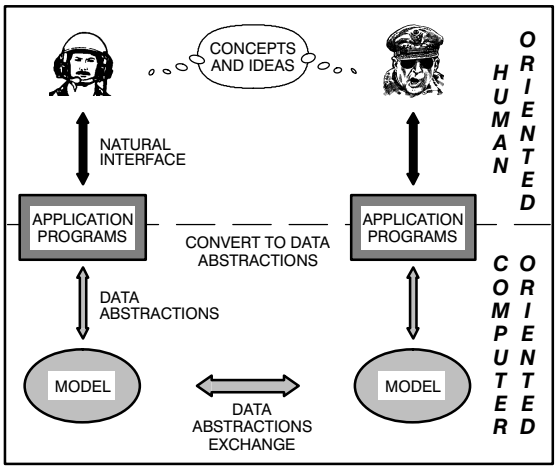

Figure 4: Applications Programs Isolate Users from Database/Data Model.

the image which may then be discarded, leaving only the symbolic sensing information in the database. For this reason, it is the data abstraction, and not the image, that is considered to be the backbone structure of the information.

A second advantage of the model−based paradigm is that it greatly facilitates interaction with other sophisticated application programs. Although the formal, mathematical form of the model precludes most users from directly accessing the database, sophisticated application programs provide a natural interface with the user. As with any good database system, the user need know nothing of the database structure. Figure 4 illustrates this flow of information.

The interface to the database is designed to facilitate use by other computer programs rather than people. This allows more expressive, efficient, and abstract query languages and facilities to be developed that work well with machines even though they would be difficult for people. It is expected that application programs always reside between the database and the user and that the application programs provide data and information visualization capabilities. Consequently, the portion of traditional query languages that support views (e.g., the *select* part of an SQL query) can be separated and delegated to the application program.

A third advantage is that the abstraction process may be generalized to the extreme. It is now plausible to ''overload'' the abstractions of military concepts to such a degree that apparent differences (encountered when building data elements through the superficial automation of manual techniques) dissolve, leaving what was originally perceived as dissimilar entities as equivalent concepts. Thus, the applications programs (that provide buffering between the user and the model) serve as a translator that converts common, overloaded abstractions into a particular vernacular or language familiar to the user.

For example, Figure 5 illustrates the common concept of a target. Figure 5a shows four different implementations of a target (e.g., the artillery calls them fire missions). In each case, there is some sort of source

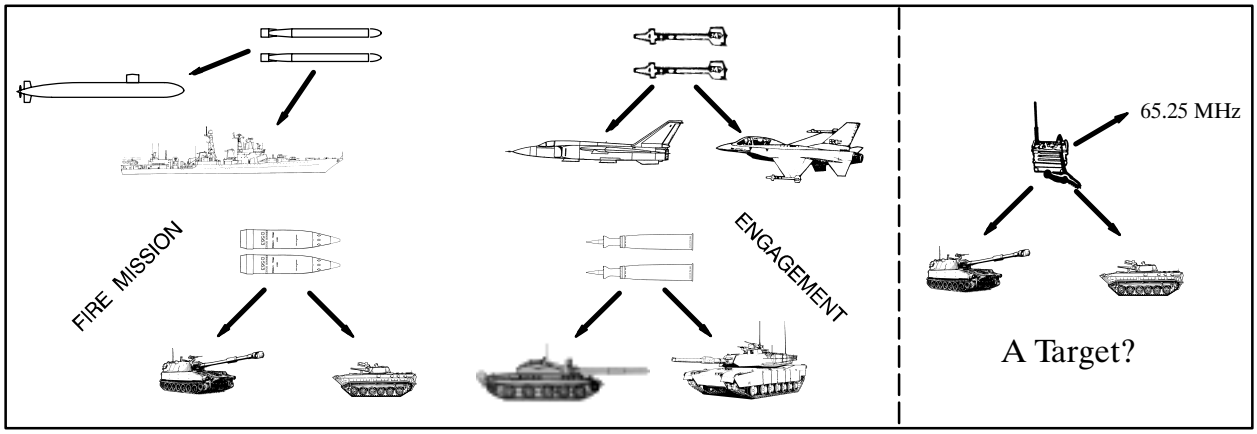

Figure 5a: The Common Concept of ''Target'' May Be Conceived Differently by Different Users.

Figure 5b: Same Basic Structure as a Target

(the shooter), a destination (the ''shootee''), some type of ordnance, and a quantity. There will be other fields, such as timing constraints, as well. Although it may not be apparent to the user, all four of these activities refer to the same general concept (what the C2 Reference Model calls the ''infliction'' interaction; see Mayk [1994]). If viewed at the ''infliction'' level of abstraction, these activities unify into the same basic data entity. Figure 5b shows another similar interaction, called a ''communications'' interaction. It too has a source, destination, ''ordnance'' (i.e., a piece of communications equipment), and a quantity (i.e., a frequency). So when viewed at the even higher ''interaction'' level of abstraction, communications reflects many of the infliction properties.[4] This process of reducing and consolidating abstractions of military concepts is the toughest challenge facing model–based battle command paradigms. It is a difficult, labor–intensive task that addresses the core of military science, but the payoffs for both interoperability and software reuse are immense, and this applies equally to both joint and coalition warfare.

A fourth advantage is realized because the information is in a form conducive to machine, not direct human, manipulation. This allows computer programming and automated ''bookkeeping'' techniques to be easily applied to the data. For example, each piece of data can be tagged with an unique identifier (i.e., an integer) that also serves as an alternate primary key (i.e., surrogate key; see Codd [1979] and Date [1986]). Each time information is entered into the database, it is assigned a *universally* unique tag; see Figure 6. Thus, equivalent tags identify equivalent semantic concepts. Therefore, textual data (e.g., a unit name or description) within replicated data items (i.e., both items have the same tag value) may be stored using different vernaculars or languages, but can still be easily identified as being the same semantic entity.

Such tags have long been part of data systems, but they have been hidden from the user. They were fundamental to network databases (see Taylor [1976][5]) and are now used in object–oriented programming and databases. The difference in this case is that the tags (i.e., pointers) are made an explicit part of the data so that they can be carefully assigned, and therefore, be made consistent across database systems. This significantly facilitates the distribution and decentralization of information across disparate databases.

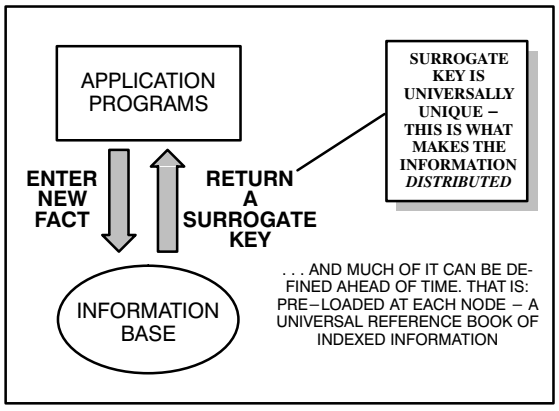

Figure 6: Universal Surrogate Keys Used As Explicit Data Identifiers.

By serving as universal pointers, explicit tags can be used to significantly improve performance. Some examples were demonstrated in an experimental prototype built by the Army Research Laboratory (see Chamberlain [1990]). In its Distributed Factbase [Hartwig, 1991], each data entry, or fact, was tagged with a surrogate key, called a fact ID.[6] Queries were designed as a two–part operation. Instead of returning data, a query returns a list of the surrogate keys (or fact IDs) of matching facts. In a second step, unbeknownst to the user, the facts are then individually retrieved.

This provided two interesting capabilities. First, a balanced binary tree was maintained that mapped fact IDs to the memory location of the corresponding fact. Given a fact ID, this allowed the facts to be retrieved in Order O(ln n) time, where n is the number of facts in the database. Queries were linear, Order O(m), where m is the number of facts of a specific type.[7] Thus, on the average, a query required Order O(mn/2) time. However, for the application program developer, many fact IDs could be cached, thus allowing many database operations to be executed in Order O(ln n) time. Second, the list of surrogate keys generated by a query can be stored within a fact using a ''list–type'' structure. This allows rapid database retrievals, reminiscent of network databases (i.e., use of indirection or following pointers), to be executed while still carefully controlling database denormalization. This controlled denormalization can be implemented by including the query that generates the list of surrogate keys as part of the schema.

Data tags also facilitate storing data across different database types. For example, data about a unit's mission could be stored in a deductive database while the spatial and temporal aspects are stored in a database optimized, or ''tuned,'' for geometric operations. The tags are used to easily maintain the cohesion between the parts of a logical entity although it may be stored in pieces based upon its different facets.

A fifth advantage is realized because each node maintains a model of the battlefield locally. This allows the amount of communications required to maintain a "common picture" to be significantly reduced (see Chamberlain [1994]). Because much of the resident information at each node is reference material, the common data tags are simply universal ''pointers'' (as are

4 The C2 Reference Model has two other types of interactions: transportation and identification.

5 Network database models have been supplanted by relational models due to their normalization characteristics.

6 A fact is equivalent to a row within a table in a relational Data Base Management System (DBMS).

7 A fact type is equivalent to a relation in a relational DBMS.

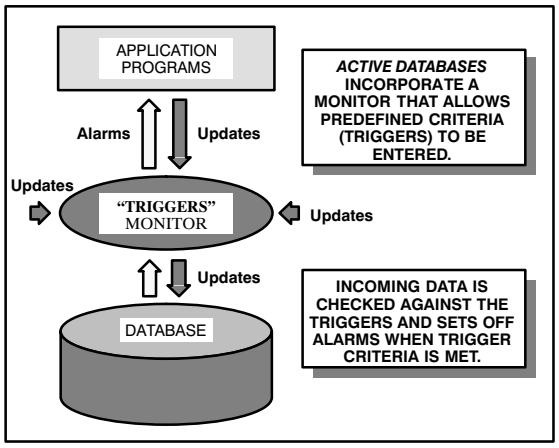

Figure 7: Active Database Structure

used in computer programs) that can be passed between databases to identify information and still maintain meaning. The tags are much more terse than the data to which they refer, and therefore, their use significantly reduces the number of ''bits'' that needs to be exchanged.

To further reduce bandwidth requirements, ''Active Database'' techniques (see Dayal [1988]; Hansen and Widom [1992]), can be used to describe synchronization requirements between models; see Figure 7.[8] The ''trigger'' concept (see Cohen [1989]) can be used to implement the synchronization criteria because, in addition to notifying the database of an event, the trigger can initiate the exchange of information; this type of trigger is called a ''distribution rule;'' see Figure 8. In other words, triggers can be used to describe how far out of synchronization the models (located at the different nodes) can become without significantly affecting military operations.

Further, by passively monitoring attached communication links, channel statistics can be maintained as part of the model. Thus, the synchronization criteria can be defined as a function of the derived communication performance, and as available bandwidth varies, so does the synchronization between databases. This allows the flow of information to be throttled automatically based upon the tactical situation that is represented and maintained within the data model.

At one extreme, that of high internodal bandwidth conditions, concurrent data models are possible; at the other extreme, when communications fails completely, the application programs can provide predictions until communications are reestablished. Of course, the interesting cases are in between these two extremes; this is illustrated in Figure 9.To maintain knowledge of the resolution of the incoming data in this varying environment, the synchronization requirements that generated the update can be easily included with the data exchange. Although potentially dubious, predictions can be a relatively trivial task if planning information is exchanged (e.g., a unit's intended objectives or routes). In any case, the system can provide some ''value–added'' even under the worse–case communication conditions; this is not easily accomplished unless a model of the battlefield is maintained at each node.

A significant result of this approach is its effect on the traditional view of database integrity. For this reason, a new style of replication mechanism is introduced, one that is ''promiscuous.''[9] Synchronization criteria may vary constantly in an attempt to provide a continuous ''best effort'' rather than a discontinuous guaranteed result. In many cases, it is simply not practical to maintain the exact same audit trail at each database; this is traditionally considered to be a gross integrity violation. In most "weak" replication mechanisms, integrity is still paramount. Updates are cached (i.e., queued) if connectivity is lost. When the connection is

8 An ''active database'' is a database where updates occur too fast to be handled by human operators. Therefore, triggers, or active queries, are used to check incoming data for specified criteria. If the criteria are met, the trigger is fired, executing some action (e.g., a signal, or alarm, is sent to the application program that entered the trigger).

9 From Webster 7th Collegiate Dictionary: Promiscuous – Casual, Irregular.

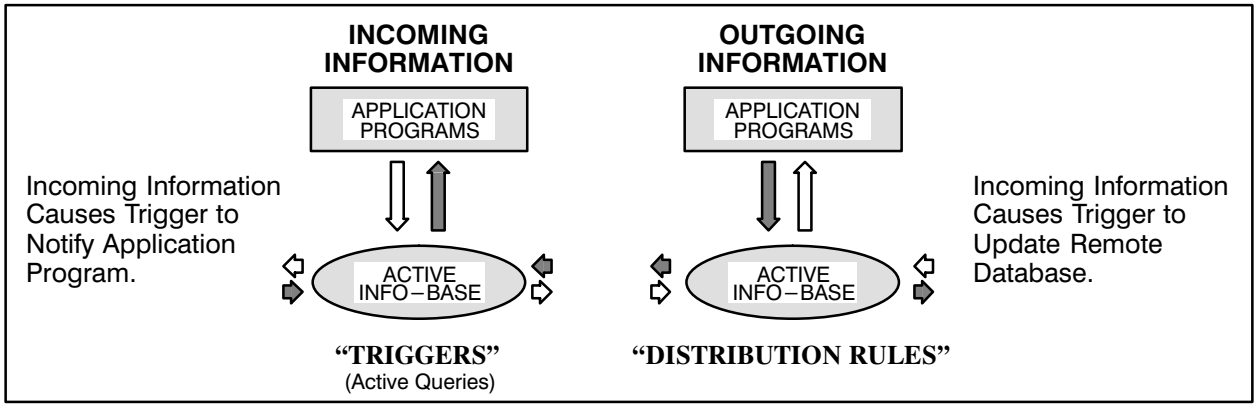

Figure 8: Active Databases Can Be Used to Control Synchronization Thresholds Between Data Models

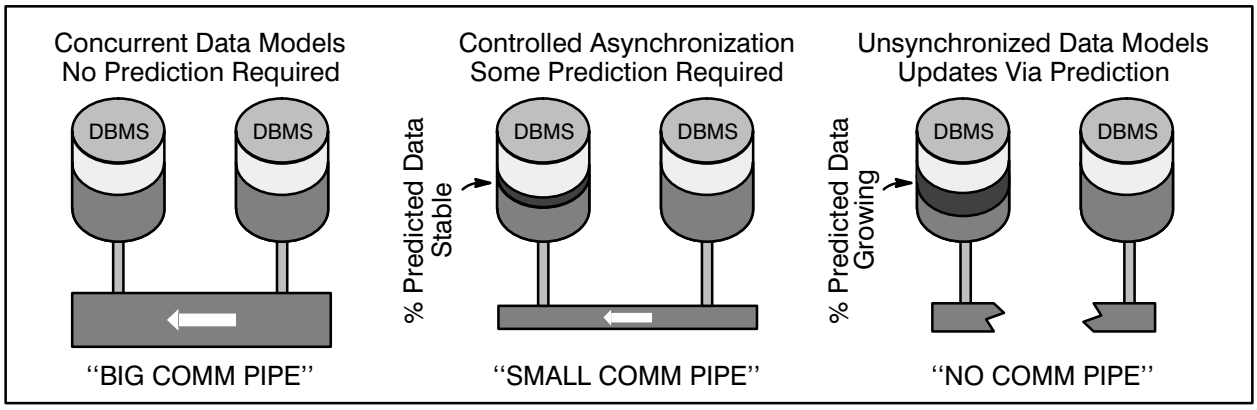

Figure 9: Synchronization Criteria Varies as a Function of Measured Bandwidth

resumed, a first−come, first−served approach is used until all the updates are applied to the receiving databases. Thus, a common audit trail (or history) is maintained at each node (i.e., the number and order of updates is consistent), with only the times associated with the actual updates being different. Using a *promiscuous* replication mechanism, one may discard ''stale'' updates from the queue so that the freshest updates can be exchanged. This is important in cases, like war, when less accurate but timely information may be more important than exact information.[10] When the situation warrants, the databases are willfully allowed to drift out of synchronization, but within known limits. Thus, this is a casual approach to database synchronization — i.e., promiscuous.

Finally, there are significant economic and interoperability advantages to the model−based paradigm. Currently, there are three ongoing, effectively duplicative, standardization domains: one for messages, one for C3I databases, and one for modeling (e.g., Distributed Interactive Simulation). Under a model−based paradigm, the data abstractions of the model define the format for data exchange (which are direct database transactions). That is, information is exchanged between databases using the same basic commands that are used to interface a database and an application program. This is the essence of Electronic Data Interchange, or EDI. Thus, ''messages'' are generated by entering information into an active database, which in turn, may generate a transaction to another database; this was illustrated back in Figure 4. This means that traditional message formats no longer need be defined to exchange abstract data because the data model representations replace the need for digital message and tactical data link (TADIL) standardization programs.[11] Traditional standard message formats are still retained for human−oriented forms of interchange (e.g., e−mail or ''freetext'' messages).

If one believes that *doing* C2 and *modeling or simulating* C2 can be based on the same set of data abstractions, then model−based battle command eliminates the requirement for many of the duplicative standardization programs in both the operational and simulation domains. This is not a unique thought. One member of the Defense Modeling & Simulation Office (DMSO) recently suggested that one think of ''operational C4 as the GUI'' (Graphical User Interface) for modeling and simulation.[12] In other words, it should be totally transparent to the user whether they are working in a real−world situation or are engaged in a simulation. The same data structures can be used for both.[13]

Using the model−based paradigm, standardization of information is done once. The purpose of communication protocols can now return to the functions of establishing procedures and packages to exchange data, regardless of its form, and the concept of a protocol data unit (or PDU), whether at the physical or application layer, can return to that of an encapsulating device rather than a data definition.

4. Conclusion

This paper discussed a different approach to building battle command systems, called *model−based* battle command. Using this approach, a formal data model of the battlefield serves as the hub of information flow. This is in contrast to the traditional message−based approach where the data model and its container, the database, are viewed as only supporting entities. Under the model−based paradigm, each node has its own, independent model of the battlefield that it maintains to the best of its ability. From this vantage point, the task of communications switches from one of exchanging messages to updating each other's databases.

10 There are anecdotes from Desert Storm about the very late arrival of highly accurate JSTARS data.

11 For example, the Army Task Force XXI defines the C2 Core Data Model as the standard database schema and the standard message set as the VMF.

12 COL Jerry Wiedewitsch at the DOD C3I Stakeholders Meeting, 2 February 1995, SWL, Inc., Vienna, VA.

13 Yet we continue to develop both the C2 Core Data Model and standard DIS PDUs.

One of the major tenets of model−based battle command is to effectively exploit the rapidly evolving computational power now available to any node in the battlefield, to include the individual warrior. At the center of the system resides an abstract data model that is maintained in a form conducive to machine manipulation. This facilitates calculations by machines to allow the insertion of true automation and decision−support facilities into battle command systems, to include sophisticated visualization capabilities.

The model−based paradigm results in several interesting effects. First, real−world operations unify with simulation into a consistent medium that can be easily traversed. Ultimately, the user can be made totally unaware of whether a simulated or real−world operation is being conducted. In other words, from a data abstraction and modeling perspective, doing C2 and simulating it are identical and the M&S community triptych of melding the real world with interactive simulations and constructive models is equally critical to the operational community; in reality, both communities have the same goals. The additional implication here is that the constructive models should provide the impetus and central focus if one wants to truly exploit rapidly evolving computer processing power.

Second, active database technology can be used to automate the exchange of information between databases. A major side effect is that information is exchanged directly between databases. The result is that conventional messages are replaced by database transactions, thus merging database and message standardization into a common task.

Third, by using appropriately abstract data, the definitions can be overloaded to accommodate multiple, functional applications across service boundaries. This significantly facilitates interoperability because the application programs that insulate the user from the database can provide the information in any vernacular, and eventually, language desired.

Fourth, because the information is stored in a computationally friendly form, it promotes the use of computationally efficient operations. A good example is information tagging in which surrogate keys are used to uniquely identify each data item. Thus, semantic equivalency can be easily traced, and voluminous, common reference material can be pre−loaded at each node.

Finally, the system is much less vulnerable to wide variations in communications bandwidth and connectivity. An automated information exchange facility can be defined for varying communication conditions. During lapses in communications, the computational power available can be used to predict data values. Better yet, by passing intended objectives and plans, expected communication problems can be nearly attenuated through prediction using a priori information, reference material, and standard operating procedures. If the communications capacity is available, it can be readily exploited. However, to provide a robust battle command system, one must base the design on the weakest link, which in most cases (especially at the lower ''fighting echelons'') is the communications system, not computational power.

References

[ATCCIS, 1993] *The NATO Army Tactical Command & Control Information System (ATCCIS) Battlefield Generic Hub Data Model*, ATCCIS Permanent Working Group, 23 April 1993.

[Bruce, 1991] Thomas A. Bruce. *Designing Quality Databases with IDEF1X Information Models*. Dorset House Publishing, New York, NY, 1991.

[Chamberlain, 1990] Sam Chamberlain. ''The Information Distribution System: IDS − An Overview,'' *BRL Technical Report BRL−TR−3114*, Jul 1990.

[Chamberlain, 1994] Sam Chamberlain. ''Automated Information Distribution in Bandwidth Constrained Environments,'' *1994 IEEE MILCOM Conference Record, Vol 2,* Oct 1994.

[Codd, 1979] E.F. Codd. ''Extending the Relational Model to Capture More Meaning,'' *ACM Transactions on Database Systems, Vol 4(4)*, Dec 1979.

[Cohen, 1989] D. Cohen. ''Compiling Complex Database Triggers,'' *Proceedings of the ACM SIGMOD*, 1989.

[Date, 1986] C.J. Dates. *Relational Databases: Selected Writings*, Addison−Wesley, Reading, MA, 1986.

[Dayal, 1988] U. Dayal. ''Active Database Management Systems,'' *Proceedings of the Third International Conference on Data & Knowledge Bases − Improving Usability & Responsiveness*, Jerusalem, Jun 1988.

[DISA, 1994] *The US DISA/JIEO Command and Control (C2) Core Data Model, Version 2*,1 DISA/JIEO/Center for Standards, ATTN: TBCE, 1 Jul 1994.

[Hansen and Widom, 1992] E.N. Hansen and J. Widom. ''An Overview of Production Rules in database Systems,'' *IBM Research Report RJ9023*, IBM Research Division, 1992.

[Hartwig, 1991] G. Hartwig, Jr., ''The Information Distribution System: The Factbase,'' BRL Technical Report BRL−TR−3247, July 1991.

[Mayk, 1994] Israel Mayk, editor. *The Command and Control Reference Model*, Basic Research Group of the JDL Tech. Panel for C3 (JDL TPC3), TR−94−1, 20 Jan 1994.

[Mayk, et al., 1993] I. Mayk, J. Kornell, and A. Fastlich. Toward C2ED_Object Libraries, In *Proceedings of the 1993 JDL Symposium on C2 Research*, National Defense University, Wash, DC., Jun 1993.

[NRad, 1994] *The Joint Warfare Simulation Object Library − Requirements Definition*, NCCOSC RDT&E Division, Code 42, 7 February 1994.

[Taylor, 1976] R.W Taylor and R.L. frank. ''CODASYL Data Base Management Systems,'' *ACM Computing Surveys, Vol 8(1)*, March 1976.

[Walker, et al., 1993] R. Walker, J. Bryden, C. Cook, G. Corliss, F. Loaiza, F. Miercort, W. Payne, F. Maggelet, E. Simaitis. *The ASDC3I/JIEO Fire Support Data Model*, IDA Paper, P−2890, 20 Sep 93.